

Express Mail Label No.

Dated: _____

Docket No.: 20193/0201145-US0
(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
Wieland Fischer et al.

Application No.: 10/825,582

Confirmation No.: 8183

Filed: April 14, 2004

Art Unit: N/A

For: **METHOD AND DEVICE FOR
CALCULATING A RESULT OF AN
EXPONENTIATION**

Examiner: Not Yet Assigned

CLAIM FOR PRIORITY AND SUBMISSION OF DOCUMENTS

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

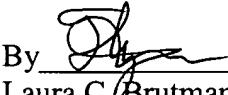
Applicant hereby claims priority under 35 U.S.C. 119 based on the following prior foreign application filed in the following foreign country on the date indicated:

Country	Application No.	Date
Germany	101 51 129.9	October 17, 2001

In support of this claim, a certified copy of the said original foreign application is filed herewith.

Dated: August 24, 2004

Respectfully submitted,

By  *Laura C. Brutman*
(53,970)

Laura C. Brutman

Registration No.: 38,395
DARBY & DARBY P.C.
P.O. Box 5257
New York, New York 10150-5257
(212) 527-7700
(212) 753-6237 (Fax)
Attorneys/Agents For Applicant

BUNDESREPUBLIK DEUTSCHLAND



Prioritätsbescheinigung über die Einreichung einer Patentanmeldung

Aktenzeichen: 101 51 129.9
Anmeldetag: 17. Oktober 2001
Anmelder/Inhaber: Infineon Technologies AG,
81669 München/DE
Bezeichnung: Verfahren und Vorrichtung zum Berechnen eines
Ergebnisses einer Exponentiation
IPC: G 06 F, H 04 L

Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.

München, den 29. April 2004
Deutsches Patent- und Markenamt

Der Präsident
im Auftrag

A handwritten signature in black ink, appearing to read "H. Agurks", is written over a stylized, wavy line that serves as a graphic flourish.

Agurks

A 9161
06/00
EDV-L

CERTIFIED COPY OF
PRIORITY DOCUMENT

PATENTANWÄLTE

European Patent Attorneys
European Trademark Attorneys

Patentanwälte · Postfach 710867 · 81458 München
Infineon Technologies AG
St.-Martin-Str. 53
81669 München

Fritz Schoppe, Dipl.-Ing.
Tankred Zimmermann, Dipl.-Ing.
Ferdinand Stöckeler, Dipl.-Ing.
Franz Zinkler, Dipl.-Ing.

Telefon/Telephone 089/790445-0
Telefax/Facsimile 089/790 22 15
Telefax/Facsimile 089/74996977
e-mail: szsz_iplaw@t-online.de

**Verfahren und Vorrichtung zum Berechnen eines Ergebnisses einer
Exponentiation**

Beschreibung

Verfahren und Vorrichtung zum Berechnen eines Ergebnisses einer Exponentiation

5

Die vorliegende Erfindung bezieht sich auf mathematische Algorithmen für kryptographische Anwendungen und insbesondere auf die Berechnung von Exponentiationen, wie sie beispielsweise im RSA-Kryptoalgorithmus verwendet werden.

10

Das RSA-Kryptosystem, das nach seinen Erfindern R. Rivest, A. Shamir und L. Adleman benannt ist, ist eines der am meisten verwendeten Public-Key-Kryptosysteme. Dieses Verfahren ist im Abschnitt 8.2 des „Handbook of Applied Cryptography“, Menezes, van Oorschot, Vanstone, CRC Press, 1996, beschrieben.

15

Das RSA-Kryptosystem kann dazu verwendet werden, um sowohl Verschlüsselungen durchzuführen als auch digitale Unterschriften auszuführen. Seine Sicherheit basiert auf der schweren Durchführbarkeit des Ganzzahlfaktorisie-

20

rungsproblems. Sowohl zur RSA-Verschlüsselung als auch zur RSA-Entschlüsselung muß eine modulare Exponentiation folgender Form durchgeführt werden:

$$E = B^d \text{ modulo } N$$

25

Hierbei stellt B die Basis dar, während d der Exponent ist und N der Modul.

30

Bei der RSA-Verschlüsselung ist der Exponent d Teil des öffentlichen Schlüssels. Bei der RSA-Entschlüsselung ist der Exponent d dagegen Teil des privaten Schlüssels, der gegen Ausspähungen gesichert werden muß.

35

Kryptographieschaltungen stehen nunmehr vor der Aufgabe, diese modulare Exponentiation einerseits sicher und andererseits schnell bzw. effizient zu berechnen. Kryptographieschaltungen werden ferner häufig in Anwendungen eingesetzt, bei denen Re-

chen- und Speicherressourcen begrenzt sind. So ist es nicht möglich, auf einer SmartCard, die z. B. für Identifikationszwecke oder in Verbindung mit Geldtransaktionen eingesetzt wird, hohe Speicher- oder Rechenressourcen vorzusehen.

5

Typischerweise wird die Exponentiation, unabhängig davon, ob sie modular ist oder nicht, mit dem sogenannten „Square-and-Multiply“-Algorithmus berechnet. Hierzu wird auf Fig. 3 Bezug genommen. Zunächst wird die Exponentiation ohne modulare Reduktion beschrieben. Im Anschluß daran wird dann ausgeführt, wie der Algorithmus im Restklassensystem des Moduls N praktiziert werden kann.

Die Aufgabe besteht darin, das Ergebnis E der Exponentiation B^d zu berechnen, wie es im Block 30 von Fig. 3 ausgeführt ist. Der Exponent d ist ein binärer Exponent und besteht aus mehreren Bits, die sich von einem höchstwertigen Bit (msb) bis zu einem niederwertigen Bit (lsb) erstrecken. Zunächst werden die Zahlen B , d bereitgestellt, wie es durch einen Block 32 in Fig. 3 dargelegt ist. Dann wird der Ergebniswert E auf einen Wert von 1 initialisiert, wie es durch einen Block 34 dargestellt ist.

Im nachfolgenden wird nun der Exponent d Stelle für Stelle untersucht bzw. abgescannt, wobei eine Stelle des Exponenten mit d_i bezeichnet ist. Hat die Stelle bzw. beispielsweise das Bit des Exponenten, das gerade untersucht wird, einen Wert von 1, wie es durch einen Entscheidungsblock 36 untersucht wird, so wird der linke Zweig in Fig. 3 eingeschlagen. Hat das untersuchte Bit des Exponenten dagegen einen Wert von 0, so wird der rechte Zweig von Fig. 3 eingeschlagen.

Wird durch den Entscheidungsblock 36 festgestellt, daß das untersuchte Bit des Exponenten einen Wert von 1 hat, so wird zunächst der Square-Schritt 38 durchgeführt, d. h. der aktuelle Ergebniswert wird quadriert. Anschließend wird in einem Multiply-Schritt 40 die Basis B auf den aktuellen Wert des

Ergebniswerts E, also das Ergebnis des Schritts 38, aufmultipliziert. In einem weiteren Entscheidungsblock 42 wird dann untersucht, ob noch weitere Stellen des Exponenten existieren. Ist dies der Fall, so wird über eine Schleife 46 zurückgesprungen und untersucht, ob die nächste Stelle d_i des Exponenten eine 1 oder eine 0 hat (Block 36). Ist die untersuchte nächste Stelle gleich Null, so wird der Square-Schritt 38' des rechten Zweigs von Fig. 3 ausgeführt. Im Gegensatz zum linken Zweig findet jedoch in dem Fall, in dem die untersuchte Stelle des Exponenten gleich 0 ist, keine Multiply-Operation, die dem Block 40 des linken Zweigs von Fig. 3 entsprechen würde, statt.

Das oben beschriebene Prozedere wird, ausgehend vom höchstwertigen Bit des Exponenten d, solange wiederholt, bis das niederwertige Bit erreicht ist. Nach der Verarbeitung des niederwertigen Bits wird der Block 42 feststellen, daß keine weiteren d_i mehr vorhanden sind. Der aktuelle Wert des Ergebniswerts E ist dann das Gesamtergebnis E der Exponentiation, das im Block 30 ausgegeben wird.

Um aus der in Fig. 3 beschriebenen Exponentiation eine modulare Exponentiation zu machen, wird im Block 32 zusätzlich zur Basis B und zum Exponenten d auch der Modul N eingegeben. Darüber hinaus findet in beiden Zweigen eine modulare Reduktion (Block 44 im linken Zweig bzw. Block 44' im rechten Zweig) statt, so daß generell gesagt nach jeder Multiplikation eine modulare Reduktion durchgeführt wird, derart, daß der Ergebniswert E am Ende der Verarbeitung für jede Stelle des Exponenten in der Restklasse des Moduls N liegt.

Es sei darauf hingewiesen, daß die Multiplikation und die modulare Reduktion nicht unbedingt in zwei aufeinanderfolgende Schritte aufgeteilt werden muß. In der Technik sind kombinierte Multiplikations-Look-Ahead- und Reduktions-Look-Ahead-Verfahren bekannt, die eine effiziente Berechnung einer Mul-

tiplikation erlauben. An dieser Stelle sei besonders der so-
genannte ZDN-Algorithmus hervorgehoben.

5 Der in Fig. 3 gezeigte Square-and-Multiply-Algorithmus in
seiner einfachsten Form ist in zweierlei Hinsicht problema-
tisch.

10 Zunächst fehlt beim Vergleich der beiden Zweige im rechten
Zweig, also wenn eine Stelle des Exponenten gleich 0 ist, ei-
ne Operation. Die beiden Zweige in Fig. 3 sind dahingehend
unsymmetrisch, daß eine Multiplikation (Block 40) ausgeführt
wird, falls eine Stelle des Exponenten gleich 1 ist, während
es im rechten Zweig keine entsprechende Operation gibt. Dies
bedeutet, daß der Square-and-Multiply-Algorithmus somit durch
15 sogenannte Timing Attacks und Power Analyses Attacks angreif-
bar ist. Um eine Homogenisierung sowohl zeitlich als auch
strommäßig herbeizuführen, d. h. daß der Zeit- und der Leis-
tungsverbrauch der Schaltung konstant sind, unabhängig davon,
ob eine 0 oder eine 1 im Exponenten steht, kann im rechten
20 Zweig eine Dummy-Multiplikation 40' eingeführt werden, wobei
das Ergebnis der Dummy-Multiplikation jedoch nicht verwendet
wird, sondern lediglich das Ergebnis des Blocks 38', also des
Square-Schritts.

25 Die Dummy-Multiplikation führt zwar zu einer zeitlichen und
strommäßigen Homogenisierung beider Zweige, erfordert jedoch
Rechenressourcen. Die Dummy-Multiplikation führt somit - auf
Kosten der Gesamtperformance der Schaltung - zu einer höheren
Sicherheit.

30 Ein weiterer Nachteil des in Fig. 3 beschriebenen Square-and-
Multiply-Algorithmus ist die Tatsache, daß dieser Algorithmus
nicht für eine parallele Ausführung geeignet ist. Wenn bei-
spielsweise der linke Zweig betrachtet wird, so ist es nicht
35 möglich, die Blöcke 38 und 40 parallel zu berechnen, da die
Berechnungen im Block 40 von den Berechnungen im Block 38 ab-
hängig sind. Ein Rechenwerk muß daher zunächst den Block 38

berechnen und dann, wenn das Ergebnis der Quadrierungsoperation vorliegt, die Berechnungen des Blocks 40, d. h. die Aufmultiplikation der Basis auf das Ergebnis des Blocks 38, durchführen.

5

Die Aufgabe der vorliegenden Erfindung besteht darin, ein sicheres und effizientes Konzept zum Berechnen eines Ergebnisses einer Exponentiation zu schaffen.

10 Diese Aufgabe wird durch ein Verfahren zum Berechnen eines Ergebnisses einer Exponentiation gemäß Patentanspruch 1 oder durch eine Vorrichtung zum Berechnen eines Ergebnisses einer Exponentiation gemäß Patentanspruch 8 gelöst.

15 Der vorliegenden Erfindung liegt die Erkenntnis zugrunde, daß von dem Square-and-Multiply-Algorithmus, der einerseits unsymmetrisch ist und andererseits lediglich eine serielle Ausführung erlaubt, weggegangen werden muß. Statt dessen wird die modulare Exponentiation unter Verwendung zweier Hilfsgrößen berechnet. Für jede Stelle des Exponenten werden zwei Multiplikationen berechnet, und zwar die Multiplikation einer Hilfsgröße mit sich selbst und die Multiplikation der beiden Hilfsgrößen miteinander. Dieses Konzept kann als Montgomery-Leiter betrachtet werden, die darauf basiert, daß der Unterschied der beiden Hilfsgrößen immer darin besteht, daß das Verhältnis der Hilfsgrößen untereinander gleich der Basis B ist.

20 Ein Vorteil der vorliegenden Erfindung besteht darin, daß unabhängig davon, welchen Wert der Exponent hat, die Anzahl der Rechenoperationen gleich ist (zwei Multiplikationen).

25 Ein weiterer Vorteil der vorliegenden Erfindung besteht darin, daß die beiden Multiplikationen, die pro Exponent zu berechnen sind, voneinander unabhängig sind. Die beiden Multiplikationen können daher parallel berechnet werden. Besonders dieses Merkmal führt zu einem Performancegewinn vom Faktor 2

gegenüber dem in Fig. 3 bezeichneten Square-and-Multiply-Algorithmus mit Dummy-Multiplikation. Selbst gegenüber dem Square-and-Multiply-Algorithmus ohne Dummy-Multiplikation wird noch ein Performancegewinn vom Faktor 1,5 durch das erfindungsgemäße Konzept erreicht.

Es sei darauf hingewiesen, daß das erfindungsgemäße Konzept inhärent sicher gegenüber Timing- und Power-Attacken ist, da die „Rechenarbeit“ des Algorithmus immer gleich ist, unabhängig davon, ob die Stelle des Exponenten gleich 0 oder gleich 1 ist.

Bei bevorzugten Ausführungsbeispielen wird das erfindungsgemäße Konzept zur Berechnung von modularen Exponentiationen eingeführt. Hierzu werden die beiden Hilfsgrößen am Ende der Verarbeitung jedes Exponenten auf die Restklasse bezüglich des Moduls N reduziert, wobei beliebige Algorithmen eingesetzt werden können, um eine Multiplikation samt modularer Reduktion unter Verwendung von Look-Ahead-Techniken durchzuführen. An dieser Stelle sei erneut auf das bekannte ZDN-Verfahren verwiesen.

Bevorzugte Ausführungsbeispiele der vorliegenden Erfindung werden nachfolgend bezugnehmend auf die beiliegenden Zeichnungen näher erläutert. Es zeigen:

Fig. 1 ein Blockschaltbild des erfindungsgemäßen Verfahrens zum Berechnen eines Ergebnisses einer Exponentiation;

Fig. 2 ein Blockdiagramm des Verfahrens von Fig. 1 mit modularer Reduktion; und

Fig. 3 eine allgemeine Darstellung des bekannten Square-and-Multiply-Algorithmus mit und ohne Dummy-Multiplikation.

In der nachfolgenden Erläuterung des erfindungsgemäßen Konzepts werden die gleichen Größen wie bei der Beschreibung von Fig. 3 verwendet. Zunächst werden in einem Schritt 100 die Basis B und der Exponent d eingegeben. In einem Schritt 102 werden dann die beiden Hilfsgrößen X und Y initialisiert, wie es in Fig. 1 dargestellt ist. Insbesondere erhält X den Wert 1, während Y den Wert der Basis B erhält. In einem Block 104 werden dann die Bits d_i des Exponenten d vorzugsweise vom höchstwertigen Bit des Exponenten sequentiell bis zum niedrigstwertigen Bit des Exponenten verarbeitet. Ist das aktuelle Bit des Exponenten d_i gleich 0, so findet die linke Spalte 104a des Blocks 104 Anwendung, während, wenn das aktuelle Bit d_i des Exponenten gleich 1 ist, die rechte Spalte 104b des Blocks 104 Anwendung findet. Insbesondere wird im Fall des Exponentenbits gleich 0 die Hilfsgröße X derart berechnet, daß sie gleich dem Quadrat der alten Hilfsgröße X ist. Die Hilfsgröße Y wird derart berechnet, daß sie gleich dem Produkt der alten Hilfsgröße X und der alten Hilfsgröße Y ist.

20 Im anderen Fall, also wenn die betrachtete Stelle des Exponenten gleich 1 ist, wird die Hilfsgröße X so berechnet, daß sie gleich dem Produkt der alten Hilfsgröße X und der alten Hilfsgröße Y ist. Die zweite Hilfsgröße Y ist gleich dem Quadrat der alten Hilfsgröße Y.

25 Aus Fig. 1 ist zu sehen, daß die beiden Produkte zur Berechnung von X und Y unabhängig voneinander sind, also parallel berechnet werden können. Diese Parallelisierbarkeit ermöglicht einen hohen Performancegewinn.

30 Wenn alle Bits d_i des Exponenten d verarbeitet sind, wird zu einem Block 106 gesprungen. Der Block 106 ermöglicht die Ausgabe des Ergebniswerts E. Der am Ende der Verarbeitung aller Bits vorhandene Wert der ersten Hilfsgröße X ist gleich dem Ergebnis der Exponentiation B^d .

35

Im nachfolgenden wird das erfindungsgemäße Konzept anhand eines einfachen Zahlenbeispiels illustriert. Es sei angenommen, daß der Exponent d binär und vierstellig ist und folgenden Wert hat:

5

$$d = 1011.$$

Nach dem Schritt 102 des Initialisierens haben die beiden Hilfsgrößen folgende Werte:

10

$$X = 1; Y = B.$$

Die höchstwertige Stelle des Exponenten d ist 1. Dies bedeutet, daß die beiden Hilfsgrößen nach einem ersten Durchlauf des Teilblocks 104b folgende Werte haben:

$$X = B; Y = B^2.$$

Die nächstniedrigere Stelle des Exponenten d ist gleich 0.

20 Dies führt dazu, daß im Block 104 der linke Teilblock 104a genommen werden muß. Am Ende der Verarbeitung durch den Block 104a haben die beiden Hilfsgrößen folgende Werte:

$$X = B^2; Y = B^3.$$

25

Die nächstniedrigere Stelle des Exponenten ist eine 1. Dies bedeutet, daß wieder der rechte Teilblock 104b des Blocks 104 durchlaufen werden muß. Am Ende der Verarbeitung haben die beiden Hilfsgrößen folgenden Wert:

30

$$X = B^5; Y = B^6.$$

Die niederstwertige Stelle des Exponenten d hat schließlich den Wert 1. Dies bedeutet, daß wieder der rechte Teilblock 104b des Blocks 104 durchlaufen werden muß. Am Ende der Verarbeitung haben die beiden Hilfsgrößen folgende Werte:

$$X = B^{11}; Y = B^{12}.$$

Nun sind alle Bits des Exponenten d verarbeitet und es kann zum Block 106 gesprungen werden. Das Ergebnis der Exponentiation, also der aktuelle Wert der Hilfsgröße X , beträgt B^{11} , wobei der Wert 11 im Dezimalsystem dem Wert 1011 im Binärsystem entspricht.

Aus dem vorangegangenen Rechenbeispiel wird deutlich, daß sämtliche Zwischenergebnisse des erfindungsgemäßen Konzepts verwendet werden, d. h. daß keine Dummy-Multiplikationen stattfinden. Sämtliche Zwischenergebnisse werden verwendet, abgesehen von dem Ergebnis der Hilfsgröße Y im letzten Schritt. Sie wird nicht benötigt und muß daher auch nicht zwingend berechnet werden. Wenn festgestellt wird, daß die letzte Stelle des Exponenten aktuell verarbeitet wird, so würde es genügen, nur die erste Hilfsgröße X zu berechnen. Aus Sicherheitsgründen kann jedoch das parallel arbeitende Rechenwerk zur Berechnung des Werts von Y ebenfalls mitlaufen. Der Ergebniswert wird dann nicht mehr verwendet.

Im nachfolgenden wird auf Fig. 2 Bezug genommen. Fig. 2 unterscheidet sich von Fig. 1 lediglich dadurch, daß statt einer üblichen Exponentiation eine modulare Exponentiation in der allgemeinen Einheitengruppe von Z modulo N berechnet wird, wie sie etwa im RSA-Verfahren benutzt wird. Hierzu findet bei jeder Berechnung der Hilfsgröße, sei es im Teilblock 104a oder im Teilblock 104b, eine modulare Reduktion mod N statt. Dies führt automatisch dazu, daß am Ende das Ergebnis der modularen Exponentiation $B^d \bmod N$ erhalten wird.

Im Pseudocode lautet der erfindungsgemäße Algorithmus folgendermaßen:

35 Eingabe: Basis B , Modul N , Exponent d

n : = Länge des Exponenten d

```
i: = 0
(X, Y): = (1, B)
während (i < n) do
  wenn  $d_i = 0$ , dann
    5   (X, Y): = ( $X^2 \bmod N$ ,  $XY \bmod N$ )
  ansonsten, wenn  $d_i = 1$ , dann
    (X, Y): = ( $XY \bmod N$ ,  $Y^2 \bmod N$ )
  ende wenn
  i: = i + 1
10  ende während
    E: = X
```

Ausgabe: E (= $B^d \bmod N$).

15 Das erfindungsgemäße Konzept ist dahingehend vorteilhaft, daß es das Zeitprofil und das Stromprofil homogenisiert und zusätzlich aufgrund einer Parallelisierbarkeit Performancegewinne erlaubt. Ferner wird kein berechnetes Zwischenergebnis verworfen. Dies wird erreicht, indem ein der Montgomery-
20 Leiter für elliptische Kurven ähnliches Konzept auf beliebige abstrakte Gruppen, wie z. B. die Einheitengruppe von $\mathbb{Z} \bmod N$ angewendet wird, wie sie etwa im RSA-Verfahren benutzt wird.

Patentansprüche

1. Verfahren zum Berechnen eines Ergebnisses E einer Exponentiation B^d , wobei B eine Basis ist und wobei d ein Exponent ist, wobei der Exponent durch eine binäre Zahl aus einer Mehrzahl von Bits darstellbar ist, mit folgenden Schritten:

5 Initialisieren (102) einer ersten Hilfsgröße X auf einen Wert von 1;

10 Initialisieren (102) einer zweiten Hilfsgröße Y auf die Basis B;

15 sequentielles Verarbeiten (104) der Bits des Exponenten durch:

20 Aktualisieren (104a) der ersten Hilfsgröße X durch X^2 oder einen von X^2 abgeleiteten Wert und Aktualisieren der zweiten Hilfsgröße Y durch $X \cdot Y$ oder einen von $X \cdot Y$ abgeleiteten Wert, falls ein Bit des Exponenten gleich 0 ist, oder

25 Aktualisieren (104b) der ersten Hilfsgröße X durch $X \cdot Y$ oder einen von $X \cdot Y$ abgeleiteten Wert und Aktualisieren der zweiten Hilfsgröße Y durch Y^2 oder einen von Y^2 abgeleiteten Wert, falls ein Bit des Exponenten gleich 1 ist; und

30 nach der sequentiellen Verarbeitung aller Bits des Exponenten, Verwenden (106) des Werts der ersten Hilfsgröße X als das Ergebnis der Exponentiation.

35 2. Verfahren gemäß Anspruch 1, bei dem in dem Schritt des sequentiellen Verarbeitens (104) von dem höchstwertigen Bit des Exponenten ausgegangen wird.

3. Verfahren gemäß Anspruch 1 oder 2,

bei dem die Exponentiation eine modulare Exponentiation $B^d \bmod N$ ist, wobei N der Modul ist, und

5 bei dem der von X^2 , XY oder Y^2 abgeleitete Wert durch modulare Reduktion mit dem Modul N von X^2 , XY bzw. Y^2 erzeugt wird.

4. Verfahren gemäß einem der Ansprüche 1 bis 3,

10 bei dem in dem Schritt des Aktualisierens, falls das Bit des Exponenten gleich 1 ist, der Wert X^2 und der Wert $X*Y$ parallel zueinander berechnet werden.

5. Verfahren gemäß einem der Ansprüche 1 bis 4,

15

bei dem in dem Schritt des Aktualisierens, falls das Bit gleich 0 ist, der Wert $X*Y$ und der Wert Y^2 parallel zueinander berechnet werden.

20 6. Verfahren gemäß einem der Ansprüche 3 bis 5,

bei dem die modulare Exponentiation bei einer RSA-Entschlüsselung und/oder einer RSA-Verschlüsselung verwendet wird.

25

7. Verfahren gemäß einem der Ansprüche 3 bis 6,

bei dem der Exponent d, die Basis B und/oder der Modul N Ganzzahlen sind.

30

8. Vorrichtung zum Berechnen eines Ergebnisses E einer Exponentiation B^d , wobei B eine Basis ist und wobei d ein Exponent ist, wobei der Exponent durch eine binäre Zahl aus einer Mehrzahl von Bits darstellbar ist, mit folgenden Merkmalen:

35

einer Einrichtung zum Initialisieren (102) einer ersten Hilfsgröße X auf einen Wert von 1;

einer Einrichtung zum Initialisieren (102) einer zweiten Hilfsgröße Y auf die Basis B;

5 einer Einrichtung zum sequentiellen Verarbeiten (104) der Bits des Exponenten durch:

10 Aktualisieren (104a) der ersten Hilfsgröße X durch X^2 oder einen von X^2 abgeleiteten Wert und Aktualisieren der zweiten Hilfsgröße Y durch $X \cdot Y$ oder einen von $X \cdot Y$ abgeleiteten Wert, falls ein Bit des Exponenten gleich 0 ist, oder

15 Aktualisieren (104b) der ersten Hilfsgröße X durch $X \cdot Y$ oder einen von $X \cdot Y$ abgeleiteten Wert und Aktualisieren der zweiten Hilfsgröße Y durch Y^2 oder einen von Y^2 abgeleiteten Wert, falls ein Bit des Exponenten gleich 1 ist; und

20 einer Einrichtung zum Verwenden (106) des Werts der ersten Hilfsgröße X als das Ergebnis der Exponentiation nach der sequentiellen Verarbeitung aller Bits des Exponenten.

9. Vorrichtung gemäß Anspruch 8,

25 bei der die Einrichtung zum sequentiellen Verarbeiten (104) ein erstes Rechenwerk und ein zweites Rechenwerk aufweist, wobei das erste Rechenwerk und das zweite Rechenwerk angeordnet sind, um parallel zueinander zu arbeiten, und

30 bei der das erste Rechenwerk angeordnet ist, um X^2 zu berechnen, falls das Bit des Exponenten 0 ist, oder $X \cdot Y$ zu berechnen, falls das Bit des Exponenten gleich 1 ist, und

35 bei der das zweite Rechenwerk angeordnet ist, um $X \cdot Y$ zu berechnen, falls das Bit gleich 0 ist, und Y^2 zu berechnen, falls das Bit gleich 1 ist.

Zusammenfassung

Verfahren und Vorrichtung zum Berechnen eines Ergebnisses einer Exponentiation

5

Zum Berechnen des Ergebnisses einer Exponentiation B^d , wobei B eine Basis ist und wobei d ein Exponent ist, der durch eine binäre Zahl aus einer Mehrzahl von Bits darstellbar ist, wird zunächst eine erste Hilfsgröße X auf einen Wert von 1 initialisiert (102). Dann wird eine zweite Hilfsgröße Y auf die Basis B initialisiert (102). Hierauf werden die Bits des Exponenten sequentiell verarbeitet (104), indem die erste Hilfsgröße X durch X^2 oder einen von X^2 abgeleiteten Wert aktualisiert wird, und indem die zweite Hilfsgröße Y durch $X \cdot Y$ oder einen von $X \cdot Y$ abgeleiteten Wert aktualisiert wird, falls ein Bit des Exponenten gleich 0 ist (104a). Falls ein Bit des Exponenten gleich 1 ist, wird die erste Hilfsgröße X durch $X \cdot Y$ oder einen von $X \cdot Y$ abgeleiteten Wert aktualisiert, und wird die zweite Hilfsgröße Y durch Y^2 oder einen von Y^2 abgeleiteten Wert aktualisiert (104b). Nach der sequentiellen Verarbeitung aller Bits des Exponenten wird der Wert der ersten Hilfsgröße X als das Ergebnis der Exponentiation verwendet (106). Damit wird eine höhere Sicherheit durch Homogenisierung des Zeit- und des Stromprofils erreicht. Außerdem wird ein Performancegewinn durch Parallelisierbarkeit möglich.

Figur 1

Figur zur Verarbeitung

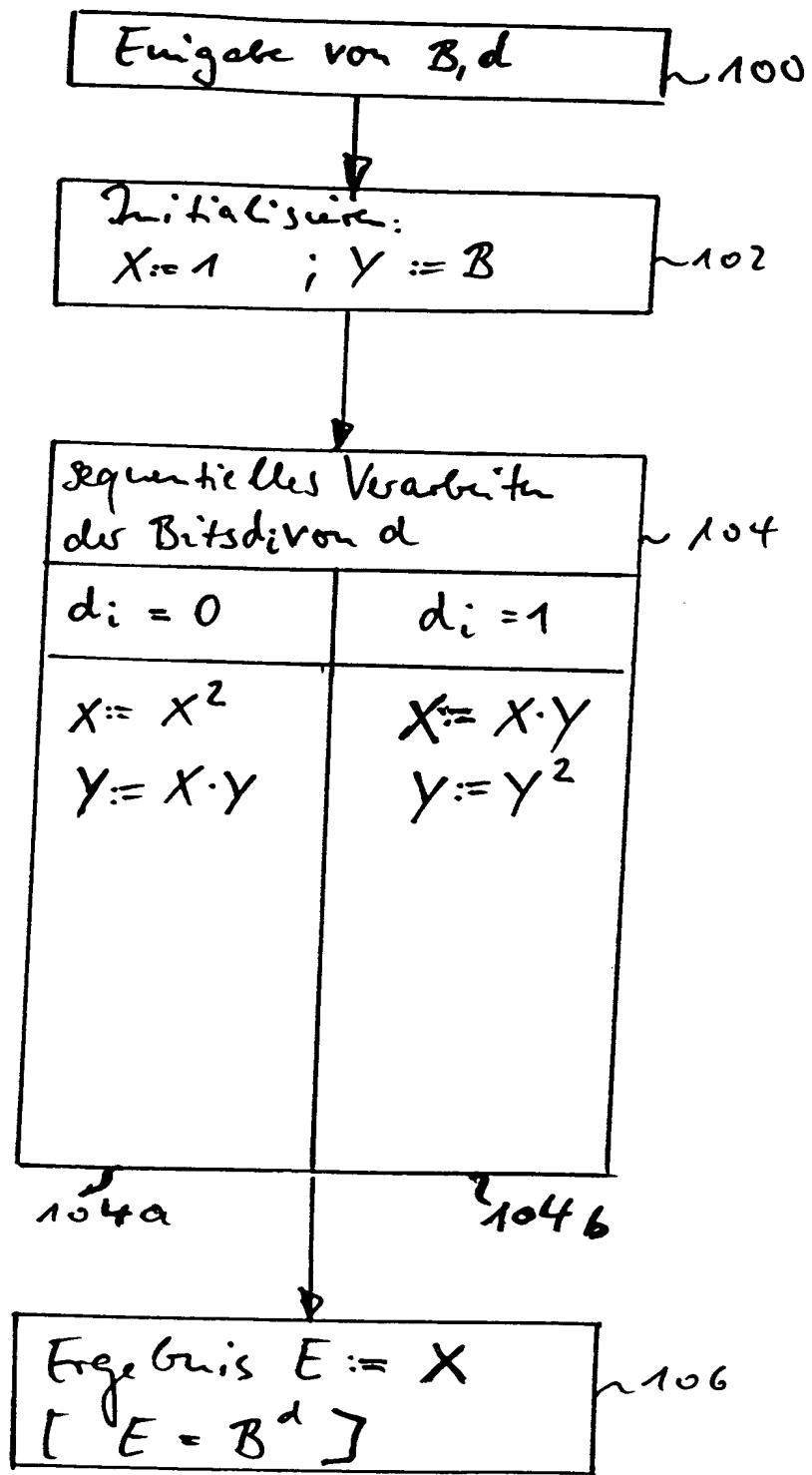


Fig. 1

Bezugszeichenliste

- 30 Ausgeben des Ergebnisses
- 32 Eingeben von B, d, N
- 34 Initialisierung des Ergebnisses
- 36 Untersuchen des Bits des Exponenten
- 38 Quadrierungsschritt, falls das Bit des Exponenten gleich 1 ist
- 38' Quadrierungsschritt, falls das Bit des Exponenten gleich 0 ist
- 40 Multiplikationsschritt
- 40' Dummy-Multiplikationsschritt
- 42 Untersuchen, ob weitere Bits d_i vorhanden sind
- 44 Reduktionsschritt, falls das Bit gleich 1 ist
- 44' Reduktionsschritt, falls das Bit des Exponenten gleich 0 ist
- 46 Iterationsschleife
- 100 Eingabe von B, d
- 102 Initialisieren von X, Y
- 104 sequentielles Verarbeiten der Bits von d
- 104a Aktualisieren der Hilfsgrößen, wenn $d_i = 0$
- 104b Aktualisieren der Hilfsgrößen, wenn $d_i = 1$
- 106 Ausgeben des Ergebnisses

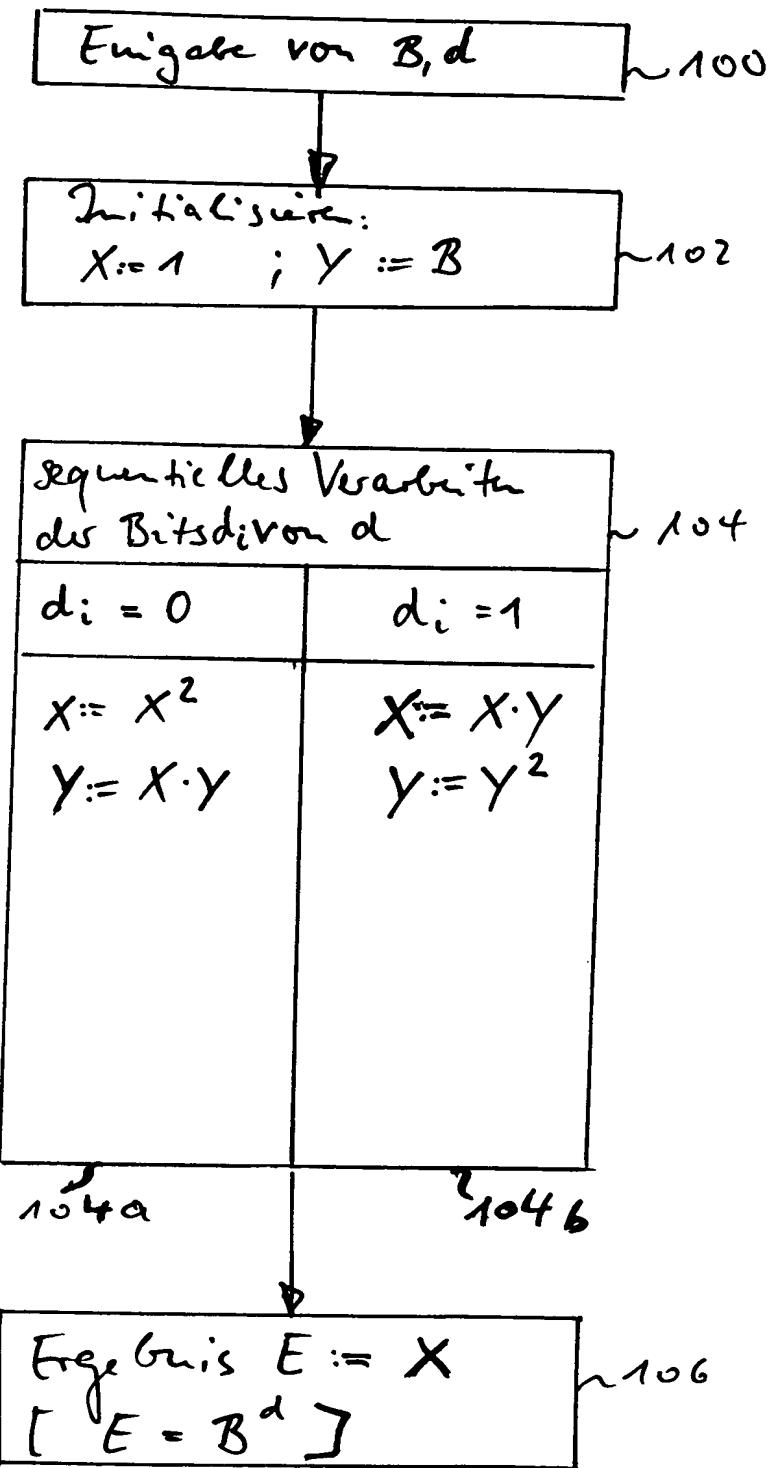


Fig. 1

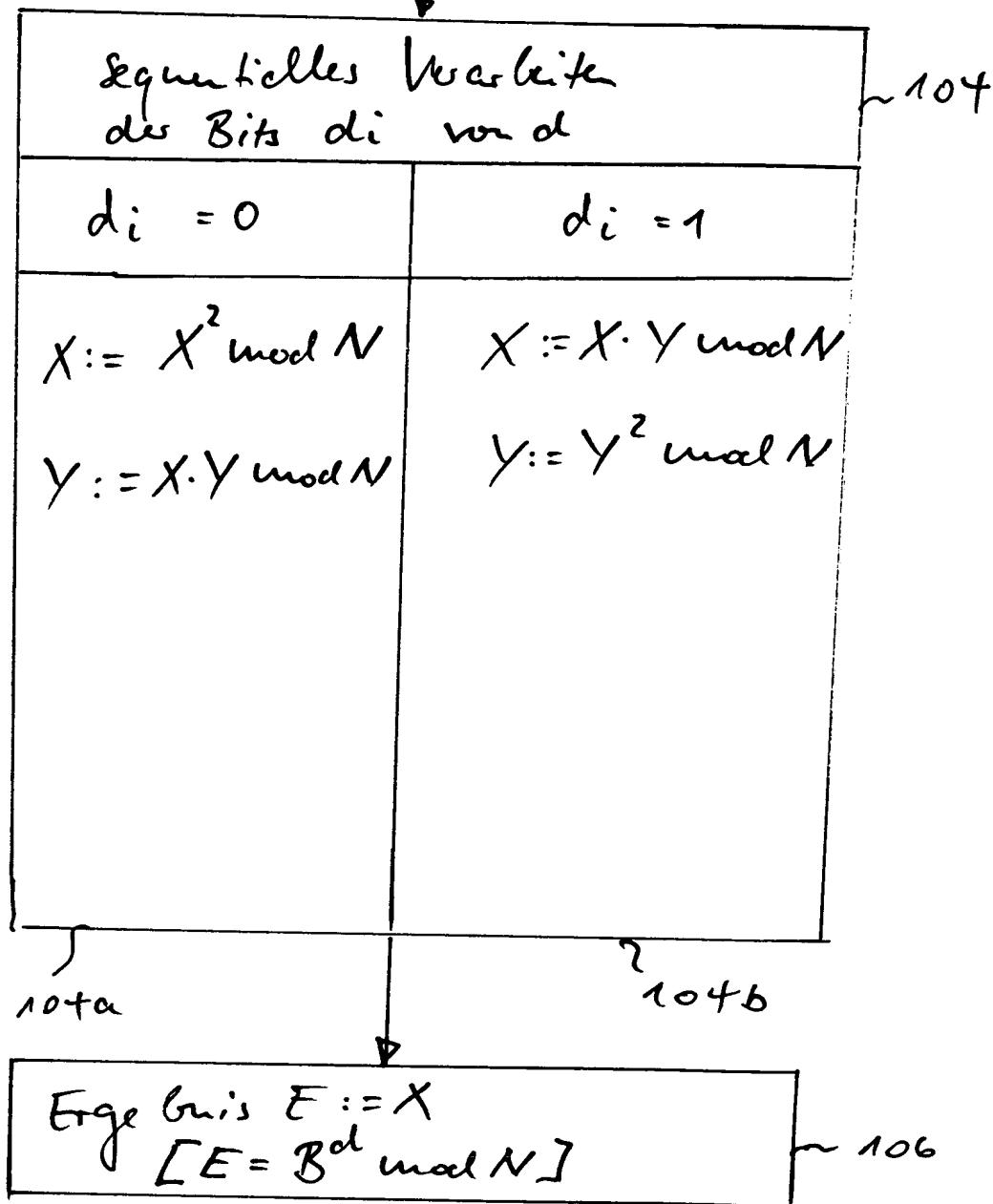


Fig. 2

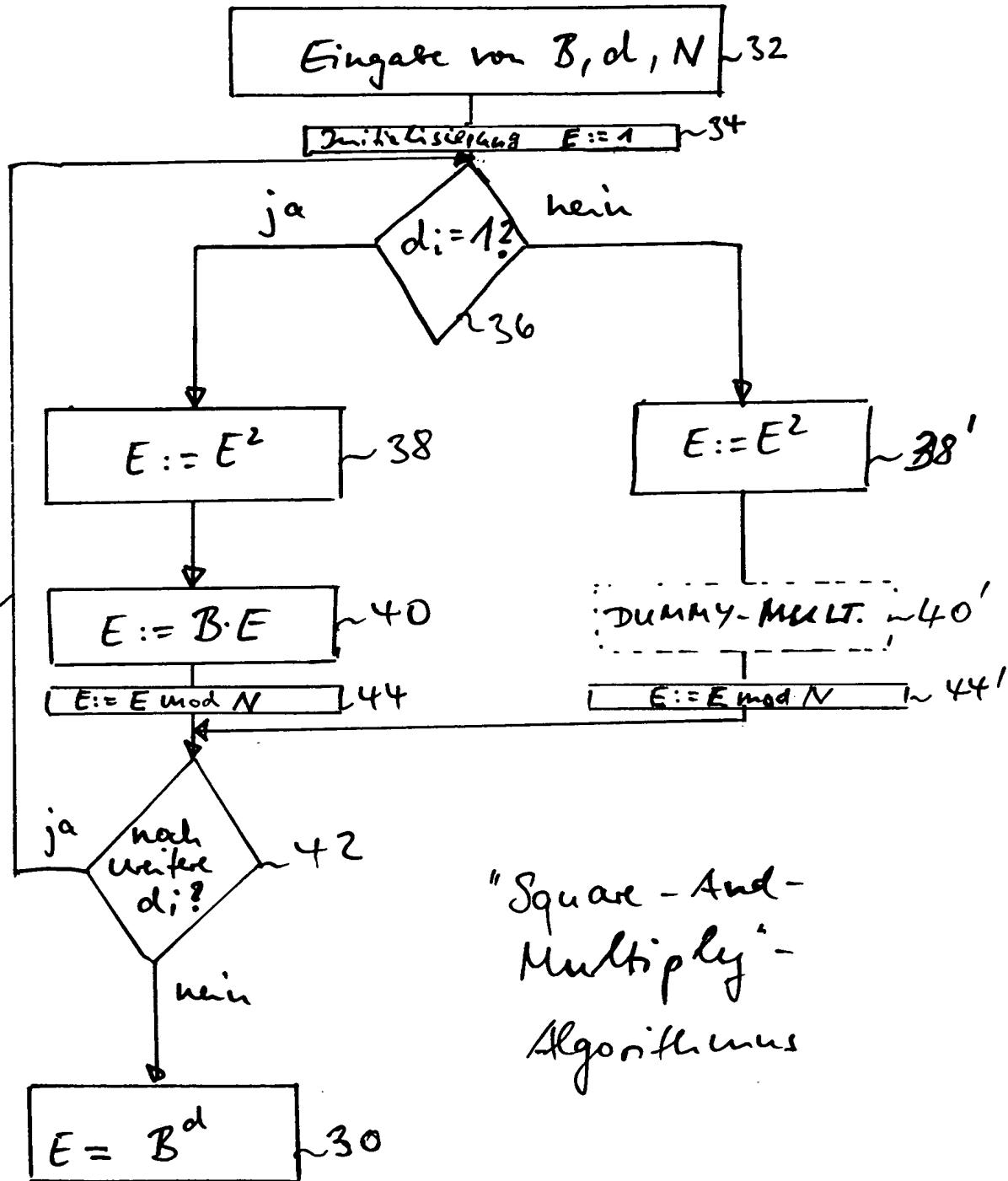


Fig. 3